//SPDX-License-Identifier: MIT

pragma solidity ^0.6.0-0.9.0;

// Write a smart contract on a test network, for Bank account of a customer for

// following operations: Deposit money | Withdraw Money | Show balance

contract Bank{

mapping(address => uint) public user\_account;

mapping(address => bool) public user\_exist;

function create\_account() public payable returns(string memory){

require(user\_exist[msg.sender] == false, "Account Already created!");

user\_account[msg.sender] = msg.value;

user\_exist[msg.sender] = true;

return "Account created";

}

function deposit(uint amount) public payable returns(string memory){

require(user\_exist[msg.sender] == true, "Account not created!");

require(amount > 0, "Amount should be greater than 0");

user\_account[msg.sender] += amount;

return "Amount deposisted sucessfully";

}

function withdraw(uint amount) public payable returns(string memory){

require(user\_exist[msg.sender] == true, "Account not created!");

require(amount > 0, "Amount should be greater than 0");

require(user\_account[msg.sender] >= amount, "Amount is greater than money deposisted");

user\_account[msg.sender] -= amount;

return "Amount withdrawn sucessfully";

}

function account\_balance() public view returns(uint){

return user\_account[msg.sender];

}

function account\_exists() public view returns(bool){

return user\_exist[msg.sender];

 }}